HARDWARE QUANTUM GATE

Cross-Reference to Related Applications

[0001] The present application is a continuation-in-part of U.S. Patent Application Serial No. 10/615,446 filed July 8, 2003, the entire contents of which are incorporated herein by reference.

Field of the Invention

[0002] The present invention relates to quantum algorithms, and more precisely, to a hardware quantum gate that performs, in particular, the entanglement operation of quantum algorithms.

Background of the Invention

[0003] Quantum algorithms are global random searching algorithms based on the principles, laws and effects of quantum mechanics. They are used for controlling a process or for processing data in a database, and more specifically, for controlling a process that may include search-of-minima intelligent operations.

[0004] In a quantum search, each design variable is represented by a finite linear superposition of initial states, with a sequence of elementary unitary steps manipulating the initial quantum state $|i\rangle$ (for the

input) such that a measurement of the final state of the system yields the correct output. Usually, three principle operators, i.e., linear superposition (coherent states), entanglement and interference, are used in the quantum search algorithm.

[0005] For a better understanding, a brief description of quantum search algorithms is provided. The problems solved by quantum algorithms may be stated as follows:

Input	A function $f:\{0,1\}^n \rightarrow \{0,1\}^m$
Problem	Find a certain property of f

The structure of a quantum algorithm is outlined by a high level representation in the schematic diagram of FIG. 1.

[0006] The input of a quantum algorithm is always a function f from binary strings into binary strings. This function is represented as a map table, which defines for every string its image. Function f is first encoded into a unitary matrix operator U_F depending on f properties. This operator calculates f when its input and output strings are encoded into canonical basis vectors of a Complex Hilbert Space: U_F maps the vector code of every string into the vector code of its image by f.

BOX 1: UNITARY MATRIX U_F

A squared matrix U_F on the complex field is *unitary* if its inverse matrix coincides with its conjugate transpose:

$$U_F{}^{-1}=U_F{}^{=}$$

A unitary matrix is always reversible and preserves the norm of vectors.

[0007] When the matrix operator U_F has been generated, it is embedded into a quantum gate G, a unitary matrix whose structure depends on the form of matrix U_F and on the problem to be solved. The quantum gate is the core of a quantum algorithm. In every quantum algorithm, the quantum gate acts on an initial canonical basis vector (the same vector can always be chosen) to generate a complex linear combination (called a superposition) of basis vectors as the output. The superposition contains all the information to answer the initial problem.

[0008] After the superposition has been created, a measurement takes place to extract this information. In quantum mechanics, measurement is a non-deterministic operation that produces as output only one of the basis vectors in the superposition. The probability of every basis vector being the output of a measurement depends on its complex coefficient (probability amplitude) in entering a complex linear combination.

[0009] The segmental action of the quantum gate and

of the measurement forms the quantum block. The quantum block is repeated k times to produce a collection of k basis vectors. In measuring a non-deterministic operation, these basic vectors would not be necessarily identical and each one of them will encode a piece of the information needed to solve the problem. The last part of the algorithm includes interpretation of the collected basis vectors to get the right answer for the initial problem with a certain probability.

[0010] The behavior of the encoder block is described in the detailed schematic diagram of FIG. 2. Function f is encoded into matrix U_F in three steps.

[0011] Step 1: The map table of function $f:\{0,1\}^n \rightarrow \{0,1\}^m$ is transformed into the map table of the injective function $F:\{0,1\}^{n+m} \rightarrow \{0,1\}^{n+m}$ such that:

$$F(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{m-1}) = (x_0, \ldots, x_{n-1}, f(x_0, \ldots, x_{n-1}) \oplus (y_0, \ldots, y_{m-1})) \quad (1)$$

BOX 2: XOR OPERATOR $\oplus$

The XOR operator between two binary strings p and q of length m is a string s of length m such that the i-th digit of s is calculated as the exclusive OR between the i-th digits of p and q:

$$p=(p_0, .., p_{n-1})$$

$$q=(q_0, .., q_{n-1})$$

$$s = p \oplus q = ((p_0+q_0) \bmod 2, .., (p_{n-1}+q_{n-1}) \bmod 2))$$

[0012] The need to deal with an injective function comes from the requirement that U_F is unitary. A unitary operator is reversible, so it cannot map two different inputs in the same output. Given that U_F is the matrix representation of F, F is supposed to be injective. If the matrix representation of function f is directly used, a non-unitary matrix could be obtained since f could be non-injective. Injectivity is thus fulfilled by increasing the number of bits and considering function F instead of function f. Function f can always be calculated from F by putting $(y_0, \ldots, y_{m-1})=(0, \ldots,0)$ in the input string and reading the last m values of the output string.

[0013] Step 2: Function F map table is transformed into U_F map table, following the following constraint:

$$\forall s \in \{0,1\}^{n+m} : U_F[\tau(s)] = \tau[F(s)] \qquad (2)$$

The code map $\tau : \{0,1\}^{n+m} \rightarrow \mathbf{C}^{2^{n+m}}$ ($\mathbf{C}^{2^{n+m}}$ is the target Complex Hilbert Space) is such that:

$$\tau(0)=\begin{pmatrix}1\\0\end{pmatrix}=|0\rangle \quad \tau(1)=\begin{pmatrix}0\\1\end{pmatrix}=|1\rangle$$
$$\tau(x_0,..,x_{n+m-1})=\tau(x_0)\otimes..\otimes\tau(x_{n+m-1})=|x_0..x_{n+m-1}\rangle \qquad (3)$$

BOX 3: VECTOR TENSOR PRODUCT ⊗

The tensor product between two vectors of dimensions h and k is a tensor product of dimension $h \cdot k$, such that:

$$|x\rangle \otimes |y\rangle = \begin{pmatrix} x_1 \\ \dots \\ x_h \end{pmatrix} \otimes \begin{pmatrix} y_1 \\ \dots \\ y_k \end{pmatrix} = \begin{pmatrix} x_1 y_1 \\ \dots \\ x_1 y_k \\ \dots \\ x_h y_1 \\ \dots \\ x_h y_k \end{pmatrix}$$

Physical interpretation:

If a component of a complex vector is interpreted as the probability amplitude of a system being in a given state (indexed by the component number), the tensor product between two vectors describes the joint probability amplitude of two systems being in a joint state.

Examples: Vector Tensor Products

$$(0,0) \xrightarrow{\tau} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle$$

$$(0,1) \xrightarrow{\tau} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle$$

$$(1,0) \xrightarrow{\tau} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |10\rangle$$

$$(1,1) \xrightarrow{\tau} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |11\rangle$$

[0014] Code τ maps bit values into complex vectors of dimension 2 belonging to the canonical basis of C^2. Besides using a tensor product, τ maps the general state of a binary string of dimension n into a vector of dimension 2^n for reducing this state to the joint state of the n bits composing the register. Every bit

state is transformed into the corresponding 2-dimensional basis vector, and then the string state is mapped into the corresponding 2^n-dimensional basis vector by composing all bit-vectors through a tensor product. The tensor product is thus the vector counterpart of a state conjunction. Basis vectors are denoted using the notation $|i\rangle$. This notation is taken from the description of quantum mechanics.

[0015] Step 3: U_F map table is transformed into U_F using the following transformation rule:

$$[U_F]_{ij} = 1 \Leftrightarrow U_F|j\rangle = |i\rangle \qquad (4)$$

which can be easily understood when considering vectors $|i\rangle$ and $|j\rangle$ as column vectors. Associating these vectors to the canonical basis, U_F defines a permutation map of the identity matrix rows. In general, row $|j\rangle$ is mapped into row $|i\rangle$. This rule will be illustrated in detail in an example quantum algorithm: Grover's algorithm.

[0016] The core of the quantum block is the quantum gate, which depends on the properties of matrix U_F. The scheme in FIG. 3 gives a more detailed description of the quantum block. The matrix operator U_F in FIG. 3 is the output of the encoder block represented in FIG. 2. Here, it becomes the input for the quantum block.

[0017] This matrix operator is first embedded into a more complex gate: the quantum gate G. Unitary matrix G is applied k times to an initial canonical basis vector $|i\rangle$ of dimension 2^{n+m}. Every time, the resulting complex superposition $G|0..01..1\rangle$ of basis vectors is measured, and one basis vector $|x_i\rangle$ is produced as a result. All

the measured basis vectors $\{|x_1\rangle, .., |x_k\rangle\}$ are collected together. This collection is the output of the quantum block.

[0018] The intelligence of such algorithms is in the ability to build a quantum gate that is able to extract the information necessary to find the required property of f and to store it into the output vector collection. The structure of the quantum gate for every quantum algorithm will be discussed in detail, observing that a general description is possible. To represent quantum some special diagrams called quantum circuits are going to be used.

[0019] An example of a quantum circuit, relative to the so called Deutsch-Jozsa's quantum algorithm, is illustrated in FIG. 4. Every rectangle is associated to a matrix $2^n \times 2^n$, where n is the number of lines entering and leaving the rectangle. For example, the rectangle marked U_F is associated to matrix U_F. Typically, matrix H represents a Hadamard rotation

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (5)$$

[0020] Quantum circuits provide a high-level description of the gate, and using some of the transformation rules listed in FIGS. 5a-5f, it is possible to compile them into the corresponding gate-matrix.

BOX 4: MATRIX TENSOR PRODUCT ⊗

The tensor product between two matrices $X_{n\times m}$ and $Y_{h\times k}$ is a (block) matrix $(n\cdot h)\times(m\cdot k)$ such that:

$$X\otimes Y=\begin{bmatrix} x_{11}Y & .. & x_{1m}Y \\ .. & .. & .. \\ x_{n1}Y & .. & x_{nm}Y \end{bmatrix} \quad with \quad X=\begin{bmatrix} x_{11} & .. & x_{1m} \\ .. & .. & .. \\ x_{n1} & .. & x_{nm} \end{bmatrix}$$

Example - Matrix Tensor Product:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\otimes\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}=\begin{bmatrix} 1\cdot\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 2\cdot\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \\ 3\cdot\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 4\cdot\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \end{bmatrix}=\begin{bmatrix} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \end{bmatrix}$$

It will be clearer how to use these rules when the first example of a quantum algorithm will be discussed in greater detail below.

[0021] The decoder block has the function to interpret the basis vectors collected after the iterated execution of the quantum block. Decoding these vectors means to retranslate them into binary strings and interpreting them directly if they already contain the answer to the starting problem. Alternatively, the vectors may be used as coefficient vectors for an equation system to get the searched solution. This part will not be discussed in any greater detail because it is relatively straightforward to understand by those skilled in the art.

[0022] Because of the particular importance of the Grover's quantum algorithm in the realization of controllers and data search algorithms in databases, a brief description of the Grover's algorithm will be given below. Grover's problem is stated as follows:

Input	A function $f:\{0,1\}^n \rightarrow \{0,1\}$ such that $\exists x \in \{0,1\}^n$: $(f(x)=1 \wedge \forall y \in \{0,1\}^n : x \neq y \Rightarrow f(y)=0)$
Problem	Find x

[0023] In the Deutsch-Jozsa's algorithm there are two classes of input functions and it must be determined what class the input function belongs to. In this case the problem is almost identical in its form, even if it is more difficult because now we are dealing with 2^n classes of input functions (each function of the kind described forms a class).

[0024] The diagram of the Grover's algorithm is depicted in FIG. 6, and the gate equation is

$$\Phi = \left[(D_n \otimes I) \cdot U_F \right]^h \cdot \left({}^{n+1}H \right) \qquad (6)$$

Operator D_n is called a diffusion matrix of order n and is responsible for interference in this algorithm. The diffusion matrix is defined as follows:

D_n	$\|0..0\rangle$	$\|0..1\rangle$	...	$\|i\rangle$	...	$\|1..0\rangle$	$\|1..1\rangle$
$\|0..0\rangle$	$\mathbf{-1+1/2^{n-1}}$	$1/2^{n-1}$	...	$1/2^{n-1}$	...	$1/2^{n-1}$	$1/2^{n-1}$
$\|0..1\rangle$	$1/2^{n-1}$	$\mathbf{-1+1/2^{n-1}}$	...	$1/2^{n-1}$	...	$1/2^{n-1}$	$1/2^{n-1}$
...	...	...	...	...	...	...	...
$\|i\rangle$	$1/2^{n-1}$	$1/2^{n-1}$	...	$\mathbf{-1+1/2^{n-1}}$	...	$1/2^{n-1}$	$1/2^{n-1}$
...	...	...	...	...	...	...	...
$\|1..0\rangle$	$1/2^{n-1}$	$1/2^{n-1}$	...	$1/2^{n-1}$	...	$\mathbf{-1+1/2^{n-1}}$	$1/2^{n-1}$
$\|1..1\rangle$	$1/2^{n-1}$	$1/2^{n-1}$	...	$1/2^{n-1}$	...	$1/2^{n-1}$	$\mathbf{-1+1/2^{n-1}}$

Grover's algorithm may be implemented in routines for searching a desired item in a set, by representing in vector form each item of the set forming an input set of vectors, and applying a Grover's algorithm to this set of vectors. The output vector represents the desired item.

[0025] The implementation of the Grover's algorithm clearly implies the calculation of several vector products. In fact, all qubits must be multiplied by matrix H, then by the entanglement matrix U_F and all qubits but the latter must be multiplied by matrix D_n.

[0026] These multiplications could be carried out via software, but it is quite evident that the number of qubits of a quantum algorithm is very critical in terms of computational speed. In fact, referring to the scheme in FIG. 6, the addition of only one qubit doubles the dimensions of the matrices. Thus, the number of elements (and products) increases exponentially.

[0027] A method of performing the superposition operation of the Grover's or Deutsch-Jozsa's quantum algorithm over an input set of vectors is disclosed in European Patent No. 1267304, which is assigned to the current assignee of the present invention. This method exploits the fact that any rotated vector, obtained performing the Hadamard rotation (on an input vector) contemplated by the superposition operation of these quantum algorithms can be easily encoded in a binary vector. Therefore, the successive tensor product of the rotated vectors for generating linear superposition vectors can be carried out by logic gates. This fact allows a noticeable time saving since logic gates are

very fast.

[0028] However, this is not sufficient to significantly speed up running these quantum algorithms because the entanglement matrix U_F is a $2^{n+1} \times 2^{n+1}$ square matrix, which implies a considerable computational weight both in the Grover's algorithm as well as in the Deutsch-Jozsa's algorithm.

[0029] Differently from other quantum algorithms, in the Grover's algorithm it is possible to iterate h times the entanglement and interference operations until the best solution is reached. An example of evolution of the Grover's algorithm with n=3 is given in FIG. 7a, in which basis vector and superposition, entanglement and interference output vectors are reported in order. Several iterations of entanglement and interference operations produce a better distribution of probability amplitudes.

[0030] Each value is a component on the output vector referred to a basis of vectors of n+1 qubits. There are couples or pairs of values having an opposite sign, referred to vectors of the basis having in common the first (leftmost) n qubits. For example, the values 0.625 and -0.625 are respectively referred to as vectors $|0110\rangle$ and $|0111\rangle$, respectively. Each pair of elements having an opposite sign represents the probability amplitude of a certain element of the database. For the considered example, the value 0.625 is the probability of the element associated to vector $|011\rangle$ after 3 iterations (h=3).

[0031] The algorithm may be iterated as far as a certain quantity is to be minimized, calculated as a function of the components of the output vector, and is

smaller than a certain pre-established value. For instance, this quantity can be the Shannon entropy:

$$S(h) = -\sum_{k=1}^{2^{n+1}} \|q_k(h)\|^2 \log\|q_k(h)\|^2 \qquad (7)$$

where $q_k(h)$ is the k-th component of the output vector Q taken after h iterations.

[0032] The components of the output vector obtained after h=15 iterations are represented in FIG. 7b. From FIG. 7b it is clear that the element of the database to be found is associated with vector $|011\rangle$, and after 15 iterations the Grover's quantum algorithm will find it with a probability of about 0.69.

[0033] From the above discussion it is evident that the problem of the large number of computations is even more crucial in the Grover's algorithm than in the Deutsch-Jozsa's algorithm because multiplication by the entanglement matrix U_F and the interference matrix $D_n \otimes I$ might be repeated many (h) times to output the best result.

Summary of the Invention

[0034] In view of the foregoing background, an object of the present invention is to provide a hardware quantum gate for running a quantum algorithms in a very fast manner.

[0035] The present invention exploits the fact that a large number of multiplications required by the entanglement operation of quantum algorithms give a null result, because only one component per row of the entanglement matrix U_F is not a null. The entanglement

operation generates an entanglement vector by permuting the places of pairs of opposite components of a linear superposition vector, depending on the value assumed by the function *f(.)*. More specifically, if function *f(.)* is null in correspondence to the vector identified by the first (leftmost) *n* qubits in common with the two *n*+1 qubit vectors to which a pair of opposite components of which the superposition vector is referred to, then the corresponding pair of components of the entanglement vector is equal to that of the superposition vector. Otherwise, it is the opposite.

[0036] Therefore, it is not necessary to calculate the entanglement matrix U_F to generate an entanglement vector from a superposition vector, but it is sufficient to copy or invert components of a superposition vector to generate corresponding components of an entanglement vector, depending on the values of the function *f(.)* processed by the quantum algorithm. This can be easily done using switches input with a pair of components having an opposite value of a superposition vector.

[0037] More precisely, the object of the present invention is to provide a quantum gate for running quantum algorithms using a certain binary function defined on a space having a basis of vectors of *n* qubits, composed of a superposition subsystem carrying out a superposition operation over components of input vectors for generating components of linear superposition vectors referred on a second basis of vectors of *n*+1 qubits. An entanglement subsystem carries out an entanglement operation over components of the linear superposition vectors for generating components of entanglement vectors. An interference

subsystem carries out an interference operation over components of the entanglement vectors for generating components of output vectors.

[0038] The entanglement subsystem comprises a command circuit generating a number (2^n) of logic command signals encoding the values of the binary function in correspondence to the vectors of the first basis, and an array of multiplexers input with the logic command signals that generate, for each superposition vector, corresponding signals representing components of an entanglement vector.

[0039] Each component of the entanglement vector referred to a respective vector of the second basis is equal to the corresponding component of the respective superposition vector if the binary function is null in correspondence to the vector of the first basis formed by the first *n* qubits of the respective vector of the second basis. Alternatively, each component may be equal to the opposite of the corresponding component of the respective superposition vector if the binary function is not a null in correspondence to the vector of the first basis formed by the first *n* qubits of the respective vector of the second basis.

Brief Description of the Drawings

[0040] The particular aspects and advantages of the invention will become more evident through the following description of several important embodiments and by referring to the attached drawings, wherein:

[0041] FIG. 1 is a block diagram of quantum algorithm in accordance with the prior art;

[0042] FIG. 2 is a block diagram of an encoder in accordance with the prior art;

[0043] FIG. 3 is a general structure of the quantum algorithm as shown in FIG. 1;

[0044] FIG. 4 is a circuit for a Deutsch-Jozsa's quantum gate in accordance with the prior art;

[0045] FIG. 5a shows an example of a tensor product transformation in accordance with the prior art;

[0046] FIG. 5b shows an example of dot product transformation in accordance with the prior art;

[0047] FIG. 5c shows the identity transformation in accordance with the prior art;

[0048] FIG. 5d shows an example of the propagation rule in accordance with the prior art;

[0049] FIG. 5e shows an example of the iteration rule in accordance with the prior art;

[0050] FIG. 5f explains the input/output tensor rule in accordance with the prior art;

[0051] FIG. 6 is an example of a circuit forming a Grover's quantum gate in accordance with the prior art;

[0052] FIGS. 7a and 7b illustrate the evolution of the Grover's quantum algorithm in accordance with the prior art;

[0053] FIG. 8 is a detailed view of entanglement and interference subsystems of the quantum gate in accordance with the present invention;

[0054] FIG. 9 is a detailed view of section I-a as shown in FIG. 8;

[0055] FIG. 10 is an embodiment of the section I-b as shown in FIG. 8;

[0056] FIG. 11 is an embodiment of an adder as shown section I-c of FIG. 8;

[0057] FIG. 12 is a detailed view of the digital section of the quantum gate in accordance with the present invention; and

[0058] FIG. 13 is one possible embodiment of the level shifters as shown in FIG. 12.

Detailed Description of the Preferred Embodiments

[0059] The quantum gate of the invention is suitable for s fast running quantum algorithm applied over a set of input vectors, such as for example, decision making or data search routines based on the Deutsch-Jozsa's algorithm or the Grover's algorithm. It is composed of a superposition subsystem carrying out a linear superposition, an entanglement subsystem carrying out an entanglement operation, and an interference subsystem carrying out an interference operation according to the quantum algorithm to be implemented.

[0060] An essential feature of the quantum gate of the invention includes the fact that the entanglement subsystem does not multiply a superposition vector for the entanglement matrix U_F, but generates components of an entanglement vector simply by copying or inverting respective components of the superposition vector depending on values of the function $f(.)$.

[0061] This allows a relevant reduction of the number of multiplications with respect to known methods, and can be carried out using multiplexers. For the sake of simplicity, the hardware quantum gate of the invention will be described in reference to the Grover's quantum algorithm for n=3, though what will be discussed may be easily repeated for other quantum algorithms (in particular, the Deutsch and the Deutsch-Jozsa's algorithm) whose entanglement operation includes a permutation of components of a superposition vector.

[0062] A hardware quantum gate of the invention

suitable for running the Grover's algorithm with any number of iterations is substantially composed of two parts. Part I: (analog) Calculation step-by-step of the output values. This part is divided in the following subsections: I-a: Entanglement; I-b and I-c: Interference. Part II: (digital) Entropy evaluation, storage of vectors for iterations and output display. This part provides also the first basis of vectors.

[0063] The analog part of this scheme for a three-qubits quantum gate is depicted in FIG. 8. A command circuit HB14 generates eight command signals Vc1, ..., Vc8 each representing a value of the function *f(.)* on a respective vector of the first basis. An array of multiplexers HB13 is input with voltage signals 011, ..., 082 representing the sixteen components of a linear superposition vector and generates signals Vo1, ..., Vo8 representing only the even or the odd components of an entanglement vector. Let us suppose that these signals represent the odd components of an entanglement vector.

[0064] Section I-a is present in every hardware quantum gate of the invention, irrespective of the quantum algorithm to be implemented. Each multiplexer, depicted in FIG. 9 is input with a pair of components (011 and 012 for example) that are referred to vectors of the second basis having the first (leftmost) 3 qubits in common ($|0000\rangle$ and $|0001\rangle$), and generates a respective component (Vo1) which is equal to 011 if Vc1 represents a null value of function *f(.)* on vector $|000\rangle$, or to 012 if Vc1 represents a non-null value.

[0065] Sections I-b and I-c depicted in FIG. 8 are specifically designed for Grover's quantum algorithms.

The presence of tensor products in the interference operation, whose number increases dramatically with the dimensions, forms a critical point.

[0066] Sections I-b and I-c of FIG. 8 of the quantum gate allow the interference operation of Grover's quantum algorithm to be quickly carried out. It has been noticed that the matrix $D_n \otimes I$ has the following properties: odd columns (or rows, because $D_n \otimes I$ is symmetric) have non-zero odd components and even columns have non-zero even components. The value of all non-zero components, except for the i^{th} component of i^{th} column (diagonal elements), is $1/2^{n-1}$. The components on the up-left down-right diagonal of the matrix differ from the other non-zero components because they are decreased by 1. The variable G^* in an entanglement vector, and the output vector of the quantum algorithm $V=(D_n \otimes I)\, G^*$ involves only a suitable weighted sum of components of G^*. The value $1/2^{n-1}$ depends only from the number n of qubits.

[0067] From the above analysis, the generic element v_i of V can be written as follows as a function of components g_i^* of the entanglement vector G^*:

$$v_i = \begin{cases} \frac{1}{2^{n-1}} \sum_{j=1}^{2^n} g^*_{2j-1} - g^*_i & \text{for } i \text{ odd} \\ \frac{1}{2^{n-1}} \sum_{j=1}^{2^n} g^*_{2j} - g^*_i & \text{for } i \text{ even} \end{cases} \qquad (16)$$

Therefore, to calculate a component v_i of the output vector it is sufficient to calculate a weighted sum of even $(\frac{1}{2^{n-1}} \sum_{j=1}^{2^n} g^*_{2j})$ or odd $(\frac{1}{2^{n-1}} \sum_{j=1}^{2^n} g^*_{2j-1})$ components of the

entanglement vector and to subtract from it the corresponding component g_i^* of the entanglement vector.

[0068] An adder HB25, depicted in greater detail in FIG. 10, sums these components with a certain scale factor that depends only on the number *n* of qubits (which is 0.25 for *n*=3), and generates a signal SQ representing the sum of the odd (or even) components of the entanglement vector. Finally, an array of adders I-c generates signals 01, ..., 08 representing the odd (or even) components of an output vector by subtracting the signals Q1, ..., Q8 from the scaled sum SQ.

[0069] A hardware quantum gate of the invention for carrying out a Deutsch-Jozsa's quantum algorithm does not have the sections I-b and I-c of FIG. 8 for performing the interference operation. In this case, the interference subsystem is substantially composed of an array of adders, each calculating a respective component 01, ..., 08 of the output vector as a linear combination of components Vo1, ..., Vo8 of a corresponding entanglement vector.

[0070] It is not necessary to calculate all components of the entanglement or output vectors, because the odd components of any vector are always opposite to the even components. For this reason entanglement and interference operations are carried out only on the odd or even components. The other components are calculated by inverting the first ones.

[0071] The adders of section I-c may be formed, for instance, as depicted in FIG. 11. A voltage of 2.5V is added to transpose the value inside the range [0÷5]V of A/D converters that interface with Part II (digital part).

[0072] To provide better performances, a different range of probability amplitudes is chosen. The presence of converters, whose range is [0÷5]V, suggests adoption of a range of [-2.5÷2.5] instead of $[-1/\sqrt{2} \div 1/\sqrt{2}]$. This implies that the output is not normalized to 1 but to $2*2.5^2=12.5$.

[0073] Because of the necessity of different types of operations (storing values, entropy evaluation and stopping iteration) for performing the Grover's algorithm, a microprocessor unit has been chosen as the core of Part II. For instance, an appropriate microprocessor unit may be the commercial device CPLD XC95288XL of STMicroelectronics, as depicted in FIG. 12.

[0074] An analog/digital converter, which may be for example the A/D converter ADC0808 of STMicroelectronics, receives signals representing components 01, ..., 08 of the output vector and produces a corresponding binary string D1, ..., D8. The microprocessor unit XC95288XL receives this string and calculates the Shannon entropy.

[0075] Summarizing, the microprocessor unit performs the following functions:

1. Drives correctly the A/D converter.
2. Acquires eight digital values and evaluates Shannon entropy S.
3. Compares S with a fixed threshold.
4. If S < threshold, stops iteration and sends the results to LED Matrix display; otherwise sends the results to a digital/analog converter.
5. Provide initial condition of superposed

basis vectors.

[0091] A display may also be connected to the CPLD to display the result. If Shannon entropy is not a minimum, the binary string has to be re-converted in an analog signal by a digital/analog converter for it to feedback into the entanglement subsystem I-a. In the preferred embodiment of FIG. 12, the digital/analog converter is the commercial device AD7228 of STMicroelectronics.

[0092] The level shifters HB3, ..., HB10, which can be formed as depicted in FIG. 13, re-translate the values of IN1, ..., IN8 into the range [-2.5÷2.5] and re-obtain all components from odd components. The iterations are carried out very fast, according to the CPLD frequency (20÷30 MHz). This fact suggests that very high performances could be reached even with a greater number of qubits, resulting in a very efficient search in a large database.